Beccy Zheng
PUI Section B
Homework 6B

Link to GitHub Repo Folder :
https://github.com/rebeccaz21/rebeccaz21.github.io/tree/master/assgn6B

Link to site: https://rebeccaz21.github.io/assgn6B/index.html

Reflection

What challenges or bugs did you encounter?

I encountered many bugs and challenges while going through this assignment. Many of these problems were able to be solved based on the in-class labs, office hours, as well as googling. I did *a lot* of googling. While the solutions are rarely ever just copy and paste, seeing other examples and code can be helpful for thinking through my own code. TA's were often also very helpful even if it was just to help me decide what to google.

First, the beginning challenges included updating the image on the product details page when users select a color. Before beginning to solve this problem, I wanted to ensure that users could only select one checkbox at a time for color and size. To solve this problem, I did a google search, something around the lines of "How to make sure only one checkbox is checked HTML Javascript". My solution was to assign each checkbox an id of a number 1-4 or 5-8. Then in my function to check that only one checkbox was selected, I looped through the list of ids. If an element with an id that was not the id that was just checked is checked, the ".checked" property would be assigned false so it would become unchecked. I then wrote another function *changePhoto*(checkbox_id, file), which will loop through the ids for color options, then change the src to the file parameter if the index is the checkbox that is checked. I realized it was easy to change the src of an image by using ".src =".

The second part of adding functionality to the page was being able to add to the user's pack (cart). This involved various tasks including: creating an object prototype for a product (cat backpack), creating an object once the user selects "Add to Pack" and filling the object with the correct information, as well as showing some visual indication of how many items were in the user's Pack. Using the lab exercises, I was able to create the product object prototype. These objects would be stored in an array in local storage. I was able to decide to use an array after talking with a TA in office hours. When users select "Add to Pack" a function *addToPack()* is called. This function will use looping and arithmetic to get which color and size is selected. Using the color, we assign the object's image. Then a new object is created and assigned values of size, color, image, and quantity. We then get the array of products from local storage, add this new object, and re-store it in local storage.

A challenge was how to update the “Pack” in the header with the number of items in the pack. I realized we could get this number from the length of the product array, and a TA (Sai) helped me during his office hours to see that we could change the innerHTML to include this number. Another challenge was how to update this number onto other pages as well. When every page loads, I ended up calling a function which will get the product array length and update the pack header text with the correct number.

Yet another challenge was being able to see all the user’s items in their cart page. This was very confusing for me because I didn’t know how to use javascript to write to HTML. A TA (Tianshi) at office hours was able to help me fix this by simply teaching me that javascript can change the innerHTML of a div using a string. Using this, I was able to loop through each object of my product array and generate the string that would create the divs for each aspect of the item.

The last challenge was being able to remove items from the cart. This was a problem because not only did the item have to be removed visually from the cart page, but also needed to be remove from the local storage array and then everything would need to be updated (the prices, and the pack number in the header). Luckily, I realized that when the cart page loads, an onLoad2() function is called which will update the HTML every time with the objects that are in the product array. Thus, we only need to be able to remove the object from the array. I used google, and talking with other students to figure out at a high-level what to do. First, I assigned each “remove” button an id that corresponds with the index of the object in the array. Then I used the *splice* built-in function to remove that specific object from the array. I then re-stored the list in local storage. Because of the functions I wrote that would be called when the cart page loaded, I knew that the visuals and numbers would already be updated in them when we removed elements from the cart.

Also, to help debug during this stage, I created a button “clear everything” which would reset the object array for the list of products to be an empty array. This way, when I wanted to test my code, I would be able to reset and follow through the user journey to ensure everything works properly. I often also used the chrome inspect feature.

What programming concepts did you learn (5 examples)?

1. *Objects*

I used object-oriented programming in order to organize how to handle the “products” that users put into their cart. I created an object template prototype named *prod* to store product color, sizes, price, and quantity. Every time a user added to their pack, an object would be created.

My object prototype:

```
function prod(size, color, image, num) {
        this.name = "cat backpack";
        this.size = size;
```

```
        this.color = color;
        this.image = image;
        this.num = num;
        this.price= 44.95;
}
```

An example of creating a new object with other variables (s, c, im):
var catbp = new prod(s,c,im,1);

2. Local Storage
In order to continue the data across pages on the website, I used local storage. I used local storage to store an array of objects. This allowed me to access the number of items in the cart for each page to visually indicate, as well as to create each item in the cart visually on the cart page with updated prices.

Example of getting the array from storage:
var currList = JSON.parse(localStorage.getItem("lstOfProd"));

Example of adding to the list and storing it back in local storage:
var test = currList.push(catbp);
localStorage.setItem("lstOfProd",JSON.stringify(currList));

3. Writing HTML using Javascript
In order to visually display product information on the Pack (cart) page, I used a javascript function to create the strings to change the innerHTML of a div that included all of the products. This way we can dynamically update the DOM and the HTML.

Example:

```
newStr = newStr+ "<div class = 'specific-prod'> <img id = 'item-1' class = 'in-cart'
src="+item.image+" alt = 'cat backpack' /> ";

newStr = newStr+"<div class = 'item-details'> <h4> Cat Backpack </h4
<h5>"+item.size+"/"+item.color+"</h5> <button class = 'remove' id = "+ i +" onclick =
'removeItem(this)'> remove </button> </div>";

newStr = newStr + "<div class = 'prod-price'> <p> $"+item.price+" </p>

document.getElementById("pack-products").innerHTML = newStr;
```

4. Using For Loops

I would often use for loops to loop through arrays, and also elements with ids that were numbers. This helped to easily access all elements in a specific category or that were grouped together. This also helped me avoid hard-coding.

Example:

```
for (var i = 5; i <= 8; i++)  {
        if (i != checkbox_id) {
                if (document.getElementById(i).checked) {
                        document.getElementById(i).checked = false;
                }

        }
}
```

5. Using Built-In Functions

Using built-in functions were often very helpful and useful. The hardest part about using them was knowing that they existed in the first place and following the syntax.

Examples:

(Using .checked)

```
if (document.getElementById(i).checked) {
        document.getElementById(i).checked = false;
}
```

(Using .toFixed)

```
var tax = Number(.07* subtotal).toFixed(2);
```

(Using splice to remove an element from an array)

```
lst.splice(index,1);
```